CS472 / CM472 / CS672 - Fall 2007: Assignment 2

Instructor: Pascal Poupart | Office: DC2514 | Email: ppoupart@cs.uwaterloo.ca
Classroom: MC4042 | Tu,Th 4:00-5:20 | Office Hours: Wed; 10:00-11:00
Web Site: www.student.cs.uwaterloo.ca/ ~cs472/ | Newsgroup: uw.cs.cs472

TA: Zhuliang Chen | Email: z4chen@cs.uwaterloo.ca
Special office hours: Monday, Oct 22, 2-3pm (DC3318)

Due: Tuesday, October 23 (at the begining of class)

In PDE-based image processing, images are treated as discrete two-dimensional functions defined on $[0,1] \times [0,1]$. Using a simple diffusion method, a noisy image, $u^0(x,y)$, can be sharpened to obtain $u^k(x,y)$ by repeated solution of the following elliptic equation (for $k \geq 0$):

$$\begin{array}{rcll} -\alpha(u^{k+1}_{xx}(x,y) + u^{k+1}_{yy}(x,y)) + u^{k+1}(x,y) & = & u^k(x,y) & \text{in } \Omega = (0,1) \times (0,1) \\ u^{k+1}(x,y) & = & 0 & \text{on } \partial\Omega, \end{array} \quad (1)$$

where $\alpha > 0$ is the parameter controlling how much noise is to be removed. If $\alpha \approx 0$, there is not much change in the image. On the other hand, if $\alpha \gg 1$, the noise as well as some details of the original image are smoothed out. Figure 1 shows a reconstructed image after 20 iterations.

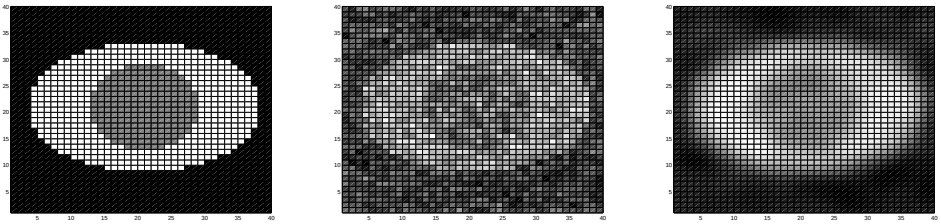


Figure 1: (Left) Original image, (Middle) Noisy image, (Right) Denoised image after 20 iterations.

Applying a standard finite difference discretization, at each grid point (x_i, y_j), we have

$$\frac{\alpha}{h^2}(4u^{k+1}_{i,j} - u^{k+1}_{i-1,j} - u^{k+1}_{i+1,j} - u^{k+1}_{i,j-1} - u^{k+1}_{i,j+1}) + u^{k+1}_{i,j} = u^k_{i,j},$$

where h is the mesh size. Thus, the linear system has the following form:

$$(\frac{\alpha}{h^2}A + I)u^{k+1} = u^k,$$

where $A =$ the standard 5-point stencil discrete Laplacian matrix, $I =$ identity matrix, and u^k, u^{k+1} are the grid functions after k and $k+1$ iterations, respectively. (Assume zero boundary condition.)

To summarize, the algorithm of the image denoising process is:

Start with u^0 (the noisy image).
for $k = 0, 1, \ldots, K$
    Solve $(\frac{\alpha}{h^2}A + I)u^{k+1} = u^k$ for u^{k+1}
end

(In this assignment, $K = 40$ and $\alpha = 3 \times 10^{-5}$.)

The objective of this assignment is to solve the image denoising problem using sparse Gaussian elimination.

The file *dirsolver.c* contains a skeleton sparse matrix solver. A symbolic factorization (analyze phase) is implemented for you in *sfac_smart*. You are to complete the other functions:

- *factor*: numeric factorization.
- *solve*: forward and back solves.

The file *matrhs.c* contains functions for setting up the image denoising problem. They are called by *main.c* to create the matrix for the image denoising equation and the initial right-hand side (which is the noisy image). It then calls the sparse solver to solve the image equation. Test your code using various grid size and orderings. A code for RCM and minimum degree orderings is provided.

Debug hints: Use a small example (e.g. 4×4) when debugging. Start by getting *factor* and *solve* to work with the natural ordering, then try to get things to work with RCM and minimum degree orderings. Once you are sure everything is OK, compile with -O for production runs. For larger grid sizes, you may want to run the program in the background and/or use the *nice* command.

1. (14 marks) Submit a listing of your code. In order to black box test your code, place your version of *dirsolver.c, dirsolver.h* in a single directory (nothing else in this directory) with the name

   ```
   your_userid_your_student_id
   ```

   Then zip up this directory using

   ```
   zip -r your_userid_your_student_id your_userid_your_student_id
   ```

   Mail the file

   ```
   your_userid_your_student_id.zip
   ```

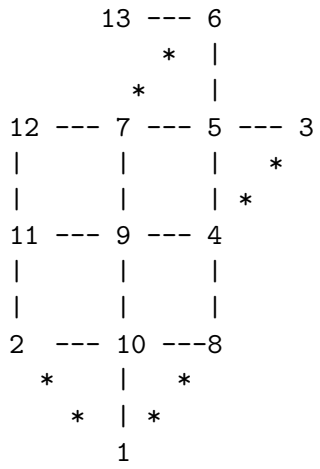
   to `z4chen@cs.uwaterloo.ca`. The total marks for this question are apportioned as

   (6 marks) Successful test of *solve* code.

   (8 marks) Successful test of your *factor* code.

2. (8 marks) Construct a table of execution times for the symbolic factor, numeric factor, and solving the image denoising problem (each phase listed separately) for the grid sizes 32×32, 64×64, 128×128 and 256×256. Also, include the number of nonzeros in the factors for each ordering and grid size. Use the natural, RCM and minimum degree orderings. On the basis of your results, experimentally determine the complexity of each phase, i.e. each phase costs $O(N^\mu)$, N =number of unknowns, estimate μ. (Hint: use MATLAB to make a loglog plot and μ is given by the slope of the graph). Comment on the results. Be sure to compile all your code with **-O** (optimization option) to get the timing results. Use the provided MATLAB program *vizimage.m* to display the denoised image. Submit the denoised images for the 4 suggested grid sizes.

3. (8 marks) Consider a matrix with symmetric structure, whose graph is given by:



Assume diagonals are selected as pivots. Give the graph of the remaining submatrix for each stage of elimination assuming that minimum degree ordering is used. Ties are broken by selecting the node with smaller original node number (as shown on the graph).

4. (10 marks) Suppose the sparse matrix A has been factored ($A = LU$) without pivoting, and that L is unit lower triangular, and U is upper triangular. Suppose the i^{th} row of $L\backslash U$ is stored in the array $af(k)$

$$\begin{aligned} i^{th}\ row\ of\ \ \mathrm{L\backslash U} &\rightarrow af(k)\ ;\ k = iaf(i), ..., iaf(i+1) - 1\ ;\ k > 0 \\ jaf(k) &\rightarrow column\ indices\ for\ element\ af(k) \\ jaf(k) &> jaf(k-1)\ \ k = iaf(i) + 1, ..., iaf(i+1) - 1 \\ jaf(\ diag(i)\ ) &= i\ ;\ diag(i) \in \{iaf(i), ..., iaf(i+1) - 1;\ diag(i) > 0\} \end{aligned}$$

Using the above arrays, describe (in words and diagrams) how you would solve

$$A^T x = b$$

Translate your algorithm into detailed pseudo code.

5. (10 marks) **Graduate student question**. Modify your *matrhs.c* code in Assignment 1 so that it will set up the 3D Laplacian matrix. (You can simply set the entries of the right-hand side vector to be 1.) Run tests using various grid sizes and ordering methods. Report the execution times in a table form. Comment on the results. Submit your modified code in *matrhs3D.c*.

Practice Questions (Do not hand in)

1. Consider the symmetric positive definite (SPD), square matrix A, where $[A]_{ij} = a_{ij}$, and where

$$a_{ij}^{(k-1)} \qquad\qquad i,j \geq k,$$

are the elements of the remaining submatrix after $(k-1)$ steps of elimination. Prove that if A is factored using the diagonals as pivots, then

$$\max_{i,j \geq k} |a_{ij}^{(k)}| \quad \leq \quad \max_{ij} |a_{ij}^{(1)}|.$$

Use the fact that for an SPD matrix, the element of largest absolute value lies on the diagonal, i.e.

$$\max_{ij} |a_{ij}| \quad = \quad |a_{mm}| \text{ for some } m.$$

What is the significance of this result for numerically factoring an SPD matrix?

2. If a symmetric matrix has a graph which is a tree, show via a few examples that no fill-in is produced with RCM ordering.

3. For nested dissection ordering, it is necessary to obtain separators (a set of nodes which divide the graph into two sets which are only connected through the nodes in the separator). It is desirable to obtain two sets of about the same size. Devise a heuristic algorithm for determining "good" separators for an arbitrary graph. (Since it's a heuristic, it may not always do a good job).

4. Consider a matrix whose graph is a three dimensional rectangular grid with nearest neighbour coupling (each interior node has six neighbours). Describe the result of RCM ordering on this three dimensional graph.